UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/620,748	07/16/2003	Mark S. Moir	6000-33800	8974

58467 7590 05/21/2009

MHKKG/SUN
P.O. BOX 398
AUSTIN, TX 78767

EXAMINER	
TSAI, SHENG JEN	
ART UNIT	**PAPER NUMBER**
2186	

NOTIFICATION DATE	DELIVERY MODE
05/21/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

patent_docketing@intprop.com
ptomhkkg@gmail.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/620,748
Filing Date: July 16, 2003
Appellant(s): MOIR ET AL.

Robert C. Kowert
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 2/23/2009 appealing from the Office action mailed 10/16/2008.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Mater

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal Identifies the ground of rejections and the associated claims under rejection to be reviewed on appeal.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

2001/0047361	Martin et al.	11-29-2001
2003/0217115	Rowlands	11/20/2003

2002/0078123 Latour 06/20/2002

(9) Grounds of Rejection

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

>>> Claims 1-4, 6-10, 12-14, 16-22, 24, 26-33 and 35-44 are rejected under 35 U.S.C. 102(a), as well as under 35 U.S.C. 102(e), as being anticipated by Martin et al. (US Patent Application Publication 2001/0047361, hereinafter referred to as Martin).

As to claim 1, Martin discloses **a computer readable storage medium encoding program code** [paragraphs 0048, 0084, 0087, 0093, 0096] **executable on one or more processors** [The present invention relates generally to coordination amongst execution sequences in a multiprocessor computer, and more particularly, to structures and techniques for facilitating non-blocking access to concurrent shared objects (paragraph 0004)] **to implement: instantiating a data structure implementation in a memory** [A computer program product encoded in at least one computer readable medium, the computer program product comprising: … (Claim 57)], **comprising**:

a double-ended array [as shown in figure 1; An important abstract data structure in computer science is the "double-ended queue" (abbreviated "deque" and pronounced "deck"), which is a linear sequence of items, usually initially empty, that supports the four operations of inserting an item at the left-hand end ("left push"), removing an item from the left-hand end ("left pop"), inserting an item at the right-hand end ("right push"), and removing an item from the right-hand end ("right pop") (paragraph 0006)];

executing a plurality of opposing-end access operations [the corresponding "opposite-end" comprises the "left hat" (figure 1, 103) and the "right hat " (figure 1, 104)] **that, when executed on the one or more processors, access the memory, and provide concurrent push-type and pop-type access to at least one of the opposing ends and concurrent** [as shown in figure 1; An important abstract data structure in computer science is the "double-ended queue" (abbreviated "deque" and pronounced "deck"), which is a linear sequence of items, usually initially empty, that supports the four operations of inserting an item at the left-hand end ("left push"), removing an item from the left-hand end ("left pop"), inserting an item at the right-hand end ("right push"), and removing an item from the right-hand end ("right pop") (paragraph 0006)], **opposing-end accesses that are non-interfering for at least some states of the array** [as shown in figure 1, the left hat and the right hat are operating on different elements located at the opposite ends, thus non-interfering with each other]**, and**

mediating concurrent execution of the access operations using a single-target synchronization primitive [The present invention relates generally to coordination

amongst execution sequences in a multiprocessor computer, and more particularly, to structures and techniques for facilitating non-blocking access to concurrent shared objects (paragraph 0004); Sometimes an implementation of such a data structure is shared among multiple concurrent processes … (paragraph 0007); Martin teaches the compare-and-swap (CAS) operation (paragraph 0012) and that CAS operations are single-target synchronization primitives (existing synchronization operations on single memory locations, such as compare-and-swap (CAS) … (paragraph 0013)]; **wherein the data structure implementation is linearizable and non-blocking** [A set of structures and techniques are described herein whereby an exemplary concurrent shared object, namely a double-ended queue (deque), is implemented. Although non-blocking, linearizable deque implementations exemplify several advantages of realizations in accordance with the present invention, the present invention is not limited thereto (paragraph 0018)], **and wherein the single-target of the single-target synchronization primitive includes a value encoding or an element of the array and a version number encoded integrally therewith** [The "compare-and-swap" operation (CAS) typically accepts three values or quantities: a memory address A, a comparison value C, and a new value N. The operation fetches and examines the contents V of memory at address A. If those contents V are equal to C, then N is stored into the memory location at address A, replacing V. Whether or not V matches C, V is returned or saved in a register for later inspection. All this is implemented in a linearizable, if not atomic, fashion. Such an operation may be notated as "CAS(A, C,

N)" (paragraph 0012); Fourth, figure 1 of Martin shows that elements of the array has a V number, such as V1, V2, V3 and son on].

As to claim 2, Martin teaches that **the storage medium of claim 1, wherein the concurrent opposing-end access operations are non-interfering for all but boundary condition states of the array** [We begin by describing the operation of "normal" push and pop operations that do not encounter any boundary cases or concurrent operations. Later, we describe special cases for these operations, interaction with concurrent operations, and operations for growing and shrinking the list (paragraph 0104)].

As to claim 3, Martin teaches that **the storage medium of claim 1, wherein the non-blocking implementation is obstruction-free, though not wait-free or lock-free** [as shown in figure 1, the left hat and the right hat are operating on different elements located at the opposite ends, thus obstruction-free; and when both of them try to access the same element, then only one may access and synchronization is needed, hence not wait-free or lock-free].

As to claim 4, Martin teaches that **the storage medium of claim 1, wherein the single-target synchronization primitive employs a Compare-And-Swap (CAS) operation** [In other realizations, a synchronization primitive such as a CAS can be used to ensure a precise count (paragraph 0123); we first use a CAS primitive to set the next node's value field to the distinguishing value RY (lines 5-7). The reason for using a CAS instead of an ordinary store is that we can determine the value overwritten when storing the RY value (paragraph 0141)].

As to claim 6, Martin teaches that **the storage medium of claim 4, wherein said mediating comprises attempting to increment the version number included in the single-target of the single-target synchronization primitive includes a value encoding for an element of the array and a version number encoded integrally therewith** [The "compare-and-swap" operation (CAS) typically accepts three values or quantities: a memory address A, a comparison value C, and a new value N (paragraph 0012); figure 1 of Martin shows that elements of the array has a V number, such as V1, V2, V3 and son on; Furthermore, although various non-blocking, linearizable deque implementations described herein employ a particular synchronization primitive, namely a double compare and swap (DCAS) operation, the present invention is not limited to DCAS-based realizations (paragraph 0019); In other realizations, a synchronization primitive such as a CAS can be used to ensure a precise count (paragraph 0123); we first use a CAS primitive to set the next node's value field to the distinguishing value RY (lines 5-7). The reason for using a CAS instead of an ordinary store is that we can determine the value overwritten when storing the RY value (paragraph 0141)].

As to claim 7, Martin teaches that **the storage medium of claim 1, wherein the double-ended array implements a deque** [An important abstract data structure in computer science is the "double-ended queue" (abbreviated "deque" and pronounced "deck") … (paragraph 0006)].

As to claim 8, Martin teaches that **the storage medium of claim 1, wherein the opposing-end access operations are at least consistent with semantics of a FIFO**

queue [Many variations, modifications, additions, and improvements are possible. For example, while various full-function deque realizations have been described in detail, realizations of other shared object data structures, including realizations that forgo some of access operations, e.g., for use as a FIFO, queue, LIFO, stack or hybrid structure, will also be appreciated by persons of ordinary skill in the art (paragraph 0153)].

As to claim 9, Martin teaches that **the storage medium of claim 1, wherein the boundary-condition states include an empty state** [Except in a special case, which is described later, two shared pointers, hereafter RHat and LHat, point to the right and left sentinels, respectively. For an empty state of the deque, left and right sentinels are adjacent. Thus, FIG. 10A depicts one representation of an empty deque (paragraph 0100)].

As to claim 10, Martin teaches that **the storage medium of claim 1, wherein the boundary-condition states include a single element state** [A DCAS failure means that either the hat has been moved by another push or pop at the same end of the queue or (in the case of a single element deque) the targeted node was popped from the opposing end of the deque. In either case, the pop_right operation loops for another attempt (paragraph 0088)].

As to claim 12, Martin teaches that **the storage medium of claim 11, wherein the boundary-condition states include a full state** [The deque is initially in the empty state (following invocation of make_deque( )), that is, has cardinality 0, and is said to have reached a full state if its cardinality is max_length_S. In general, for deque

implementations described herein, cardinality is unbounded except by limitations (if any) of an underlying storage allocator (paragraph 0054)].

As to claim 13, Martin teaches that **the storage medium of claim 11, wherein the opposing-end accesses include opposing-end, push-type accesses; and wherein the boundary-condition states include a nearly full state** [An important abstract data structure in computer science is the "double-ended queue" (abbreviated "deque" and pronounced "deck"), which is a linear sequence of items, usually initially empty, that supports the four operations of inserting an item at the left-hand end ("left push"), removing an item from the left-hand end ("left pop"), inserting an item at the right-hand end ("right push"), and removing an item from the right-hand end ("right pop") (paragraph 0006); Ideally, operations on one end of the deque would never impede operations on the other end of the deque unless the deque were nearly empty (containing two items or fewer) or, in some implementations, nearly full (paragraph 0009)].

As to claim 14, Martin teaches that **the storage medium of claim 1, wherein distinct left null and right null distinguishing values are employed to identify free elements of the array** [as shown in figure 1, the left hat and the right hat are operating on different elements located at the opposite ends, thus indicating free elements; if both of them are pointing to the same element, then it is an indication that there is no free element].

As to claim 16, Martin teaches that **the storage medium of claim 1, embodied as a software component combinable with program code to provide the program**

code with non-blocking access to a concurrent shared object [A concurrent shared object representation encoded in one or more computer readable media, the concurrent shared object representation comprising: … (Claim 56)].

As to claim 17, Martin teaches that **the storage medium of claim 1, embodied as a program executable to provide non-blocking access to a concurrent shared object** [Concurrent Shared Object Implemented Using a Linked-List with Amortized Node Allocation (title); Although non-blocking, linearizable deque implementations exemplify several advantages of realizations in accordance with the present invention, the present invention is not limited thereto (paragraph 0018); Claims 56-57].

As to claim 18, Martin teaches **the storage medium of claim 1, comprising at least one medium selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium** [The computer program product of 57, wherein the at least one computer readable medium is selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium (Claim 62)].

As to claim 19, it recites substantially the same limitations as in claim 1, and is rejected for the same reasons set forth in the analysis of claim 1. Refer to "As to claim 1" presented earlier in this Office Action for details.

As to claim 20, it recites substantially the same limitations as in claim 3, and is rejected for the same reasons set forth in the analysis of claim 3. Refer to "As to claim 3" presented earlier in this Office Action for details.

As to claim 21, it recites substantially the same limitations as in claim 2, and is

rejected for the same reasons set forth in the analysis of claim 2. Refer to "As to claim 2" presented earlier in this Office Action for details.

As to claim 22, it recites substantially the same limitations as in claim 6, and is rejected for the same reasons set forth in the analysis of claim 6. Refer to "As to claim 6" presented earlier in this Office Action for details.

As to claim 24, it recites substantially the same limitations as in claim 4, and is rejected for the same reasons set forth in the analysis of claim 4. Refer to "As to claim 4" presented earlier in this Office Action for details.

As to claim 26, Martin teaches **the storage medium of claim 19, wherein at least some concurrently executed access operations interfere with each other; and wherein the interfering concurrently executed access operations are each retried** [as shown in figure 1, the left hat and the right hat are operating on different elements located at the opposite ends, thus obstruction-free; and when both of them try to access the same element, then only one may access and synchronization is needed; If one or more other executions of push_right operations intervene and consume the newly allocated nodes, this retry behavior will again note the shortage and again call upon add_right_nodes to allocate more nodes until eventually there is at least one (paragraph 0085)].

As to claim 27, Martin teaches **the storage medium of claim 26, wherein the non-blocking deque implementation does not guarantee that at least one of the interfering concurrently executed access operations makes progress** [as shown in figure 1, the left hat and the right hat are operating on different elements located at

the opposite ends, thus obstruction-free; and when both of them try to access the same element, then synchronization is needed, hence not wait-free or lock-free].

As to claim 28, Martin teaches **the storage medium of claim 27, wherein a separate contention management facility is employed to ensure progress in a concurrent computation that employs the deque implementation** [Concurrent Shared Object Implemented Using a Linked-List with Amortized Node Allocation (title); Furthermore, although various non-blocking, linearizable deque implementations described herein employ a particular synchronization primitive, namely a double compare and swap (DCAS) operation, the present invention is not limited to DCAS-based realizations (paragraph 0019); paragraph 0108].

As to claim 29, refer it recites substantially the same limitations as in claim 1, and is rejected for the same reasons set forth in the analysis of claim 1. Refer to "As to claim 1" presented earlier in this Office Action for details.

As to claim 30, it recites substantially the same limitations as in claim 1, and is rejected for the same reasons set forth in the analysis of claim 1. Refer to "As to claim 1" presented earlier in this Office Action for details.

As to claim 31, it recites substantially the same limitations as in claim 2, and is rejected for the same reasons set forth in the analysis of claim 2. Refer to "As to claim 2" presented earlier in this Office Action for details.

As to claim 32, it recites substantially the same limitations as in claim 3, and is rejected for the same reasons set forth in the analysis of claim 3. Refer to "As to claim 3" presented earlier in this Office Action for details.

As to claim 33, it recites substantially the same limitations as in claim 4, and is rejected for the same reasons set forth in the analysis of claim 4. Refer to "As to claim 4" presented earlier in this Office Action for details.

As to claim 35, it recites substantially the same limitations as in claim 1, and is rejected for the same reasons set forth in the analysis of claim 1. Refer to "As to claim 1" presented earlier in this Office Action for details.

As to claim 36, it recites substantially the same limitations as in claim 28, and is rejected for the same reasons set forth in the analysis of claim 28. Refer to "As to claim 28" presented earlier in this Office Action for details.

As to claim 37, it recites substantially the same limitations as in claim 28, and is rejected for the same reasons set forth in the analysis of claim 28. Refer to "As to claim 28" presented earlier in this Office Action for details. Further, figures 1-14 of Martin illustrate the changing occurring the operations.

As to claim 38, it recites substantially the same limitations as in claim 28, and is rejected for the same reasons set forth in the analysis of claim 28. Refer to "As to claim 28" presented earlier in this Office Action for details.

As to claim 39, refer it recites substantially the same limitations as in claim 28, and is rejected for the same reasons set forth in the analysis of claim 28. Refer to "As to claim 28" presented earlier in this Office Action for details.

As to claim 40, it recites substantially the same limitations as in claim 1, and is rejected for the same reasons set forth in the analysis of claim 1. Refer to "As to claim 1" presented earlier in this Office Action for details.

As to claim 41, it recites substantially the same limitations as in claim 27, and is rejected for the same reasons set forth in the analysis of claim 27. Refer to "As to claim 27" presented earlier in this Office Action for details.

As to claim 42, it recites substantially the same limitations as in claim 28, and is rejected for the same reasons set forth in the analysis of claim 28. Refer to "As to claim 28" presented earlier in this Office Action for details.

As to claim 43, it recites substantially the same limitations as in claim 1, and is rejected for the same reasons set forth in the analysis of claim 1. Refer to "As to claim 1" presented earlier in this Office Action for details.

As to claim 44, it recites substantially the same limitations as in claim 28, and is rejected for the same reasons set forth in the analysis of claim 28. Refer to "As to claim 28" presented earlier in this Office Action for details.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

>>> Claims 5, 25 and 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Martin et al. (US Patent Application Publication 2001/0047361, hereinafter referred to as Martin), and in view of Rowlands (US Patent Application Publication 2003/0217115).

Regarding claims 5, 25 and 34, Martin does not teach the single-target synchronization primitive employs a Load-Linked (LL) and Store-Conditional (SC) operation pair.

However, Rowlands teaches this limitation in the invention "Load-Linked/Store Conditional Mechanism in a CC-NUMA System," where the LL/SC operation pair is used for synchronization in a multiprocessor system [paragraphs 0003-0008].

Rowlands also teaches that the motivation of using LL/SC as a synchronization mechanism is because it allows other processors to access the memory location for which the atomic read-modify-write is being attempted, which is not permitted in other synchronization mechanisms such as atomic read-modify-write [With the load-linked/store conditional mechanism, other processors may access the memory location for which the atomic read-modify-write is being attempted. If a modification occurs, the load-linked/store conditional sequence is repeated. When the store conditional completes successfully, an atomic read-modify-write of the location has been performed (paragraph 0008)].

Therefore, it would have been obvious for one of ordinary skills in the art at the time of Applicants' invention to use LL/SC as a synchronization mechanism, as demonstrated by Rowlands, and to incorporate it into the existing scheme disclosed by Martin, in order to allows other processors to access the memory location for which the atomic read-modify-write is being attempted, which enhances the overall throughput of the system.

>>> Claims 11, 15 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Martin et al. (US Patent Application Publication 2001/0047361, hereinafter referred to as Martin), and in view of Latour (US Patent Application Publication 2002/0078123).

Regarding claims 11, 15 and 23, Martin does not teach the array is indexable as a circular array.

However, Latour teaches this limitation in the invention "Method and Apparatus for Resource Access Synchronization" [The sequence of mutexes can be held in an array, a ring buffer, a linked list, a circular linked list, or any other suitable data structure that enables the order to be preserved (paragraph 0024); FIG. 9 illustrates the storage approach used in preferred embodiment of the invention. This uses a circular linked list, which is a combination of a bi-directional linked list and a ring buffer (paragraph 0054)].

Latour also teaches that the motivation of using a circular linked list is because it provides more flexibility than other types of storages [The use of the circular linked list buffer provides flexibility in that the size of the ring can readily be changed. In this manner, the number of storage locations in the ring can readily be changed to take account of changing circumstances, while still exercising control over the memory required successfully, an atomic read-modify-write of the location has been performed (paragraph 0054)].

Therefore, it would have been obvious for one of ordinary skills in the art at the time of Applicants' invention to use a circular linked list, as demonstrated by Latour, and

to incorporate it into the existing scheme disclosed by Martin, in order to provide more flexibility.

(10) Response to Arguments

Appellants' arguments have been fully and carefully considered with Examiner's answers set forth below.

Answer to Arguments on Claims 1, 4, 7-10, 12, 16-18, 24, 26, 29-30, 33, 35, 40 and 43

Appellants contend, regarding claim 1, that the Martin reference fails to teach the limitation of "wherein the single-target of the single-target synchronization primitive includes a value encoding for an element of the array and a version number encoded integrally therewith," because the Examiner's reference to figure 1 of Martin showing that elements of the array has a V number, such as V1, V2, V3 and son on, does not teach the cited limitations. The Examiner disagrees.

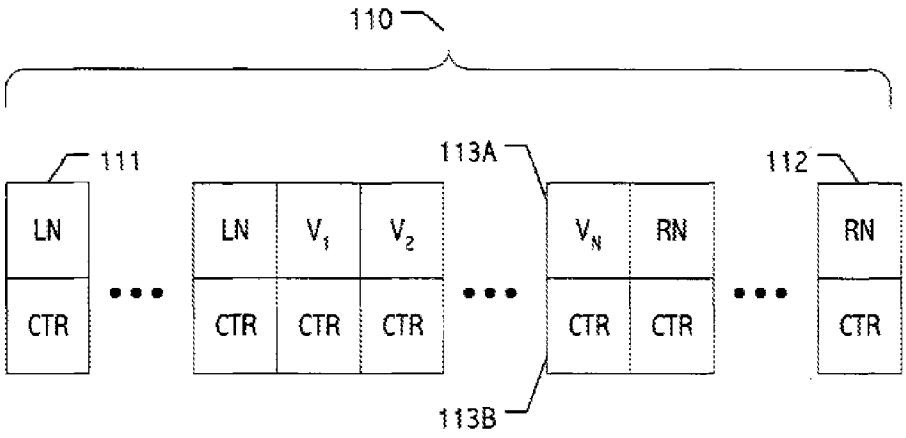
First, the Examiner interprets the limitation "a value encoding for an element of the array" as "a value-encoding mechanism/operation to encode a value for an element of the array." Note that the wording is "value encoding" and not "value encoded." A "value encoding" is a mechanism/operation of encoding a value while a "value encoded" is a value has is encoded using the recited mechanism.

Second, the Examiner interprets the limitation "a version number encoded integrally therewith" as "it is a version number that is the value that is encoded by the value-encoding mechanism/operation recited in the earlier part of the limitation."

Third, figure 1 of the Martin clears shows that each element of the array is encoded with a value of “V1,” “V2,” “V3,” …etc. In fact, Appellants acknowledge in their argument that Martin describes the V numbers of FIG. 1 as “Values are represented as “V1,” “V2,” etc …” (see the last paragraph, page 12 of the Appeal Brief). Thus, Martin clearly teaches the limitation of “a value encoding process/operation,” as values are encoded into each element of the array shown in figure 1 as a “value,” and it is also clear that “V1,” “V2,” “V3,” …etc. are the values that are encoded using the encoding process.

Fourth, as for the argument that “V1,” “V2,” and “V3" not being a “version number,” it is noted that the entire claim mentions the element “a version number” only once, and is completely silent regarding the scope and what constitutes a “version number,” (for example, a version with respect to what attribute) and how the version number is related to the other limitations recited in the claim.

Further, as shown below, figure 1 of Appellants' drawing illustrates their invention comprising an array with the version numbers denoted as "V1," "V2," … "Vn."



Likewise, as shown below, figure 1 of Martin's drawing illustrates their invention comprising an array with the corresponding version numbers denoted as "V1," "V2," … "Vn."

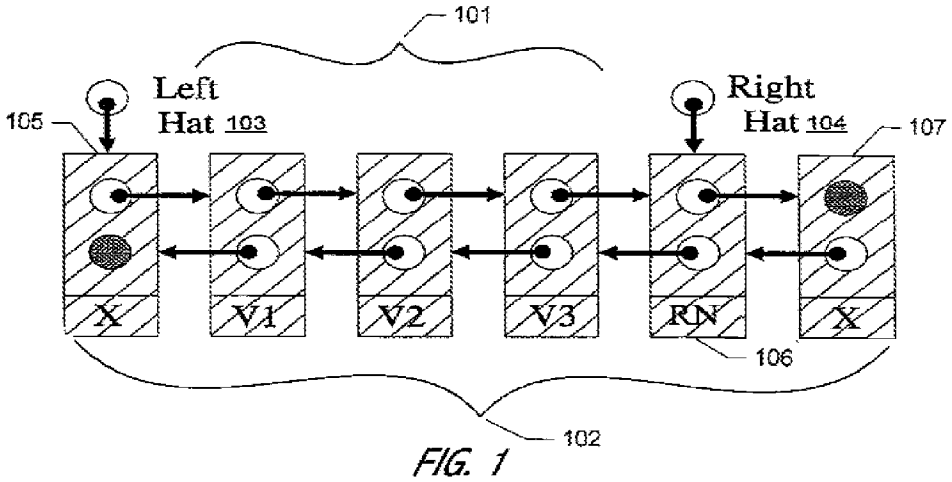


FIG. 1

The similarity between the respective figure 1 of Appellants' drawing and Martin's drawing is evident. Even the notation of "V1," "V2," … "Vn" are identical.

Moreover, according to The American Heritage College Dictionary, the word "version" may be defined as "an account from one point of view as opposed to another," and Martin teaches in paragraph [0012] that V stores different versions of the values of N as a result of the "compare-and-swap" (CAS) operations [..., a new value N ... If those contents V are equal to C, then N is stored into the memory location at address A, replacing V (paragraph 0012)]. Hence, V represents a particular version of N values associated with each CAS operation, i.e., from the point of view of one CAS as opposed to another CAS.

Thus, under the broadest, reasonably interpretation, the element "a version number" may be considered as "a value" that is associated with each element of the array, representing a particular version of N values associated with each CAS operation.

Therefore, Martin teaches the cited limitation.

Answer to Arguments on Claims 2, 21 and 31

Appellants contend, regarding claim 2, that the Martin reference fails to teach the limitation of "wherein the concurrent opposing-end access operations are non-interfering for all but boundary condition states of the array." The Examiner disagrees.

First, figure 1 of Martin clearly shows concurrent opposing-end access operations, denoted as "Left Hat" and "Right Hat," respectively.

Second, Martin teaches [the left and right ends do not interfere with each other until there is one or fewer items in the queue (abstract)]. Martin clearly teaches that the

two operations are non-interfering for all but when there is one or fewer items in the queue, which is the corresponding boundary condition state.

Thus, Martin clearly teaches this particular limitation.

Answer to Arguments on Claims 3, 20 and 32

Appellants contend, regarding claim 3, that the Martin reference fails to teach the limitation of "wherein the non-blocking implementation is obstruction-free, though not wait-free or lock-free." The Examiner disagrees.

First, Appellants define the term "obstruction-free" as "A synchronization technique is obstruction-free if it guarantees progress for any thread that eventually executes in isolation" (see paragraph [1006] of Appellants' Specification).

Second, Martin teaches [the left and right ends do not interfere with each other until there is one or fewer items in the queue (abstract)], and figures 1-14 of Martin clearly illustrate that the left and right operations are able to make progress advancing along the array until trying to access the same element of the array, which is when interference occurs. Thus, Martin teaches that it guarantees progress for any thread that eventually executes in isolation without interference, hence "obstruction-free."

Therefore, Martin clearly teaches this particular limitation.

Answer to Arguments on Claim 6

Appellants contend, regarding claim 6, that the Martin reference fails to teach the limitation recited in claim 6 due to the element of "version number" as in the case of claim 1.

The argument of the element “version number” has been answered in the portion of “***Answer to Arguments on Claims 1, 4, 7-10, 12, 16-18, 24, 26, 29-30, 33, 35, 40 and 43***.” Refer to the portion for details.

As for the argument that DCAS is not a single–target synchronization primitive, Martin teaches [The Hat Trick deque requires only a single DCAS for most pushes and pops (abstract)].

Further, the claims never define the scope and what constitute a “target,” thus allowing the interpretation that a complex-target comprising multiple simpler sub-targets to be treated as a unit of “single” target. For example, a double-word variable may be treated as a single target of operation even though it contains two words.

Answer to Arguments on Claim 13

Appellants contend, regarding claim 13, that the Martin reference fails to teach the limitation “wherein the opposing-end accesses including opposing-end, push-type accesses; and the boundary-condition states include a nearly full state.” The Examiner disagrees.

As for claim 13, Martin teaches [An important abstract data structure in computer science is the "double-ended queue" (abbreviated "deque" and pronounced "deck"), which is a linear sequence of items, usually initially empty, that supports the four operations of inserting an item at the left-hand end ("left push"), removing an item from the left-hand end ("left pop"), inserting an item at the right-hand end ("right push"), and removing an item from the right-hand end ("right pop") (paragraph 0006)], and [Ideally, operations on one end of the deque would never impede operations on the other end of

the deque unless the deque were nearly empty (containing two items or fewer) or, in some implementations, nearly full (paragraph 0009)].

As for the argument of the limitation "a circular array," this particular limitation is taught by a secondary reference (Latour) as presented in "As to claim 11." Refer to the analysis provided in "As to claim 11" for details.

Answer to Arguments on Claim 14

Appellants contend, regarding claim 14, that the Martin reference fails to teach the limitation "wherein distinct left null and right null distinguishing values are employed to identify free elements of the array." The Examiner disagrees.

It is noted that, in addition to the Examiner's explanation using figure 1 of Martin, Martin also explicitly teaches [Each node has a left pointer to its left neighbor and a right pointer to its right neighbor. The doubly-linked chain is terminated at its end nodes by a null in the right pointer of the rightmost node and a null in the left pointer of the leftmost node (paragraph 0074)].

Thus, Martin clearly teaches the cited limitation.

Answer to Arguments on Claim 19

Appellants contend, regarding claim 19, that the Martin reference fails to teach the limitation "wherein shared storage usage of the deque implementation is insensitive to a number of access operations that concurrently access the deque," because this particular limitation is not included in the limitations of claim 1 and the Examiner fails to address it. The Examiner disagrees.

It is noted that in the analysis for claim 1 (refer to "As to claim 1" for details), the Examiner cites the "deque" implementation and operation as an instance of showing why the teaching from Martin reads on the limitations recited in claim 1; and the Examiner also cites "push" and "pop" operations illustrated in figure 1, which are the corresponding "a number of accessing operations," as another instance of showing why the teaching from Martin reads on the limitations recited in claim 1. Thus, for at least these two instances, the particular limitation recited in claim 19 is also taught.

Thus, Martin clearly teaches the cited limitation.

Answer to Arguments on Claim 22

Appellants contend, regarding claim 22, that the Martin reference fails to teach the limitation "wherein state of the deque is encoded using an array," because this particular limitation is not included in the limitations of claim 6 and the Examiner fails to address it. The Examiner disagrees.

It is noted that claim 6 depends from claim 1, hence the claim analysis for claim 1 automatically applies to claim 6 as well since claim 6 inherits all limitations recited in claim 1.

It is also noted that in the analysis for claim 1 (refer to "As to claim 1" for details), the Examiner cites the "deque" implementation and operation as an instance of showing why the teaching from Martin reads on the limitations recited in claim 1; and the Examiner also cites the array illustrated in figure 1 as an example of the deque implementation and operations.

It is further noted that the limitation "version number" recited in claim 6 may be reasonably interpreted as the limitation "state" recited in claim 22, because claim 22 is completely silent on the scope and what constitutes "a state."

Thus, Martin clearly teaches the cited limitation.

Answer to Arguments on Claims 27 and 41

Appellants contend, regarding claims 27 and 41, that the Martin reference fails to teach the limitation "wherein the non-blocking deque implementation does not guarantee that at least one of the interfering concurrently executed access operations makes progress" for the same reason why the Martin reference fails to teach the "obstruction-free" limitation recited in claim 3.

The Examiner has fully addressed this issue in the portion of "***Answer to Arguments on Claims 3, 20 and 32***" presented earlier in this section. Refer to that portion for detailed explanation.

Answer to Arguments on Claims 28, 37, 38, 39, 42 and 44

Appellants contend, regarding claims 27 and 41, that the Martin reference fails to teach the limitation "wherein a separate contention management facility is employed to ensure progress in a concurrent computation that employs the deque implementation," because DCAS is part of the deque implementation. The Examiner disagrees.

It is noted that Martin teaches [The Hat Trick deque requires only a single DCAS for most pushes and pops. The left and right ends do not interfere with each other until there is one or fewer items in the queue, and then a DCAS adjudicates between competing pops (abstract)].

First, it is noted that normally when the left and right ends do not interfere with each other the DCAS is not even included in the operation, and that the DCAS is used only when there is one or fewer items in the queue. Thus, DCAS is absent, hence separated, from the operations, most of the time when there is no interference, and only included when there is one or fewer items in the queue.

Second, when there is one or fewer items in the queue the DCAS is used to adjudicates between competing pops, hence a contention management facility.

Thus, Martin clearly teaches the cited limitation.

Answer to Arguments on second ground of rejections

Appellants contend, regarding claim 5, that the Rowlands reference fails to teach the limitation "mediating concurrent execution of the access operations using a single-target synchronization primitive; and wherein the single-target of the single-target synchronization primitive includes a value encoding for an element of the array and a version number encoded integrally therewith."

Answer to Arguments part 1.

First, it is noted that the cited limitation is taught by the primary reference – Martin, as explained in the portion "***Answer to Arguments on Claims 1, 4, 7-10, 12, 16-18, 24, 26, 29-30, 33, 35, 40 and 43***." Refer to the portion for details.

Second, the Rowlands reference is relied upon to teach the particular synchronization primitive of LL/SC as recited in claim 5.

Third, since this is a 103(a) rejection for claim 5, it is not required that either one of the references to teach all the limitations recited in claim 5. It is suffices that the

combination of the Martin and Rowlands teaches all the limitations recited in claim 5. And it is clear that the combination of Martin and Rowlands teaches all the limitations recited in claim 5, as presented in "As to claim 5."

Answer to Arguments part 2.

As to Appellants' argument that the Examiner fails to provide a valid reason to combine the two references, it is noted:

First, both references are directed toward synchronization mechanism/primitive for multiple accesses competing for the same memory resource [Martin: concurrent shared object implemented using a linked-list with amortized node allocation (title); abstract; paragraph 0009; Rowlands: This invention is related to processors and, more particularly, to synchronization mechanisms for multiprocessor systems (paragraph 0003)]. Thus the two references clearly share the same inventive subject matter of synchronization mechanism/primitive for multiple accesses competing for the same memory resource.

Second, the Examiner has cited explicitly in the previous Office Action (refer to claim analysis for claim 5) at least one motivation why Rowlands' teaching of LL/SC synchronization mechanism is advantageous, as it allows other processors to access the memory location for which the atomic read-modify-write is being attempted, which is not permitted in other synchronization mechanisms such as atomic read-modify-write [Rowlands: paragraph 0008].

Therefore, the combination of Martin and Rowlands is proper and well motivated.

Answer to Arguments on third ground of rejections

Appellants contend, regarding claim 11, that the Latour reference fails to teach the limitation "wherein the array is indexable as a circular array," because Latour teaches the use of locking mechanism.

Answer to Arguments part 1.

It is noted that a circular array is a general, commonly used data structure in the art that is equally applicable to schemes that may or may not use locks. Thus the fact that Latour may have used locks in their invention does not change the fact that Latour teaches using an indexable circular array.

It is further noted that Appellants' claim 3 recites "wherein the non-blocking implementation is obstruction-free, though not wait-free or lock-free." Thus, it is clear that Appellants' claimed limitation does not require lock-free conditions. Thus, the fact that Latour may have used locks in their invention is totally irrelevant to the limitations associated with claim 11.

Answer to Arguments part 2.

As to Appellants' argument that the Examiner fails to provide a valid reason to combine the two references, it is noted:

First, both references are directed toward synchronization mechanism/primitive for multiple accesses competing for the same memory resource [Martin: concurrent shared object implemented using a linked-list with amortized node allocation (title); abstract; paragraph 0009; Latour: Method and Apparatus for resource Access Synchronization (title); abstract]. Thus the two references clearly share the same

inventive subject matter of synchronization mechanism/primitive for multiple accesses competing for the same memory resource.

Second, the Examiner has cited explicitly in the previous Office Action (refer to claim analysis for claim 5) at least one motivation why Latours' teaching of using a circular array is advantageous, as it allows the size of the array to readily be changed [Latour: The use of the circular linked list buffer provides flexibility in that the size of the ring can readily be changed. In this manner, the number of storage locations in the ring can readily be changed to take account of changing circumstances, while still exercising control over the memory required successfully, an atomic read-modify-write of the location has been performed (paragraph 0054)].

Therefore, the combination of Martin and Latour is proper and well motivated.

(11) Related Proceedings Appendix

None.

/Sheng-Jen Tsai/ Primary Examiner, Art Unit 2186
/Matt Kim/ Supervisory Patent Examiner, Art Unit 2186
/Kevin L Ellis/ Acting SPE of Art Unit 2187

May 18, 2009